com.saba.doe

Class AccessibilityPicker

```
java.lang.Object
  |
  +--com.saba.denver.SabaObject
        |
        +--com.saba.doe.AccessibilityPicker
```

public class **AccessibilityPicker**
extends SabaObject

AccessibilityPicker is a custom class that will manage the data for the DOE/SFA/Accenture Accessibility needes. This class is designed to work in compliance with the *SFA University Modernization LMS Project - Phase II, LMS Accessibility Issues Design, 17 December, 2001*

The accessibility will be stored in several relational tables in the Saba database. All IDs generated per the Accessibility requirements in any table will be incrementally numbered, have appropriate prefixes (see FGT_DD_CLASS for list) for data dictionary compliance and be numbered from 98 * 10^13 and higher.

The list of the tables are:

- FGT_GEN - a generic entity relationship (ER) table that relates one saba object to another. Each ID attribute will hereby be prefixed by the 5 character `acces' string. For relating accessibility needs to the order, the ID1 attribute will contain an order ID from TPT_OE_ORDER, while the ID2 attribute will contain a list of values ID to FGT_LIST_OF_VAL. For storing comments per *each* of the forementioned accessibility needs, the ID1 attribute will contain the list of values ID and the STR1 attribute will contain the descriptive free-form field possibly filled in by the user. Please note STR *can* and *may* be NULL; the ID2 attribute will contain (for consistency's and redundancy's sake) the order id.

- FGT_LOV - location to hold list-of-values (LOVs). Each row in this table represents one abstracted list of values by name. Each item in FGT_LOV is stored in:

- FGT_LIST_OF_VAL - holds each item of the named list of values from the FGT_LOV table. The LIST_ID contains a reference to the FGT_LOV.ID attribute earlier described.

- TPT_OE_ORDER - is the order table in which all order information is stored. The CUSTOM0 attribute in this table golds the general comment possibly entered by the learner when registering for any given order.

Constructor Summary

AccessibilityPicker(java.lang.String siteName, java.lang.String orderId) Instantiate a new AccessibilityPicker.

Method Summary

java.lang.Object[]	**getAccessibilityIds**() Obtain a set of Saba database IDs representing a list of values for accessibility needs.
java.lang.String	**getAccessibilityName**(java.lang.String id) Per each obtainable accessibility IDs, obtain the user interface name for the id.
void	**insert**(java.lang.Object[] accessibilityIds) Update the order with the given set (array) of accessibilityIds.
static void	**main**(java.lang.String[] argv) Unit test method.
void	**setAccessibilityComment**(java.lang.String accessibilityId, java.lang.String comment) Set the comment in the FGT_GEN.STR1 attribute per the given accessibilityId.
void	**setComment**(java.lang.String comment) setComment() will set (update) the comment and the TPT_OE_ORDER.CUSTOM0 attribute per the named order.

Methods inherited from class com.saba.denver.SabaObject
close, finalize, getConnection, log

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

AccessibilityPicker

```
public AccessibilityPicker(java.lang.String siteName,
                           java.lang.String orderId)
                    throws com.saba.exception.SabaException
```

Instantiate a new AccessibilityPicker.

Parameters:

`siteName` - is a valid name of a SabaSite. This is by default `SabaWeb' but can be of any name depending on what is named in SabaAdmin. This parameter can be obtained from the getSiteName() method in a SabaPage.

`orderId` - is the order id attribute for which we will be attaching data.

Method Detail

setComment

```
public void setComment(java.lang.String comment)
                throws com.saba.exception.SabaException,
                       java.sql.SQLException
```

setComment() will set (update) the comment and the TPT_OE_ORDER.CUSTOM0 attribute per the named order.

Parameters:

`comment` - is the comment string that will be placed in this field.

---

setAccessibilityComment

```
public void setAccessibilityComment(java.lang.String accessibilityId,
                                    java.lang.String comment)
                             throws com.saba.exception.SabaException,
                                    java.sql.SQLException
```

Set the comment in the FGT_GEN.STR1 attribute per the given accessibilityId.

Parameters:

`accessibilityId` - is one of the FGT_LIST_OF_VAL.IDs that can be obtained through the getAccessibilityIds() method.

`comment` - is the comment to which to attach to the given accessibility need.

---

insert

```
public void insert(java.lang.Object[] accessibilityIds)
            throws com.saba.exception.SabaException,
                   java.sql.SQLException
```

Update the order with the given set (array) of accessibilityIds.

Parameters:

`accessibilityIds` - should be a distinct set of FGT_LIST_OF_VAL.IDs that represent those accessibility needs that are identified by the user from the order_receipt.saba page.

---

getAccessibilityIds

```
public java.lang.Object[] getAccessibilityIds()
                                       throws com.saba.exception.SabaException,
                                              java.sql.SQLException
```

Obtain a set of Saba database IDs representing a list of values for accessibility needs.

Returns:

An array of String's that represent the set values and distinct FGT_LIST_OF_VAL.IDs in the database. These are **not the selected accessibilityIds, but rather all possibilities listed in the database for accessibility needs.**

---

getAccessibilityName

```
public java.lang.String getAccessibilityName(java.lang.String id)
                                       throws com.saba.exception.SabaException,
                                              java.sql.SQLException
```

Per each obtainable accessibility IDs, obtain the user interface name for the id.

Parameters:

`id` - **The accessibility id.**

Returns:

The name associated with the id.

---

main

```
public static void main(java.lang.String[] argv)
                 throws java.lang.Throwable
```

Unit test method.

```
/* $Id:$ */

/* Copyright (C) 2002   Saba Software, Services, J. S. Jensen
 *    mailto:jsjensen@saba.com
 * All rights are non-exclusive.
 */

package com.saba.doe;

import java.sql.*;
import java.util.*;

import com.saba.db.*;
import com.saba.exception.*;

/** AccessibilityPicker is a custom class that will manage the
  * data for the DOE/SFA/Accenture Accessibility needes.
  * This class is designed to work in compliance with the
  * <i>SFA University Modernization LMS Project - Phase II,
  *   LMS Accessibility Issues Design, 17 December, 2001 </i>
  * <p>
  * The accessibility will be stored in several relational tables in
  * the Saba database.  All IDs generated per the Accessibility
  * requirements in any table will be incrementally numbered, have
  * appropriate prefixes (see FGT_DD_CLASS for list) for data
  * dictionary compliance and be numbered from 98 * 10^13 and higher.
  * <p> The list of the tables are: <p>
  * <li> FGT_GEN - a generic entity relationship (ER) table that
  *   relates one saba object to another.  Each ID attribute
  *   will hereby be prefixed by the 5 character `acces' string.
  *   For relating accessibility needs to the order, the ID1 attribute
  *   will contain an order ID from TPT_OE_ORDER, while the ID2
attribute
  *   will contain a list of values ID to FGT_LIST_OF_VAL.
  *   For storing comments per <i>each</i> of the forementioned
  *   accessibility needs, the ID1 attribute will contain the list of
  *   values ID and the STR1 attribute will contain the descriptive
  *   free-form field possibly filled in by the user.  Please note
  *   STR <i>can</i> and <i>may</i> be NULL; the ID2 attribute will
contain
  *   (for consistency's and redundancy's sake) the order id.<p>
  * <li> FGT_LOV - location to hold list-of-values (LOVs).  Each
  *   row in this table represents one abstracted list of values
  *   by name.  Each item in FGT_LOV is stored in:<p>
  * <li> FGT_LIST_OF_VAL - holds each item of the named list of values
  *   from the FGT_LOV table.  The LIST_ID contains a reference to the
  *   FGT_LOV.ID attribute earlier described.<p>
  * <li> TPT_OE_ORDER - is the order table in which all order
information
  *   is stored.  The CUSTOM0 attribute in this table golds the general
  *   comment possibly entered by the learner when registering for any
  *   given order.
  * @author jsjensen@saba.com
  */
public class AccessibilityPicker extends com.saba.denver.SabaObject {

      private long id;
      private String orderId;

      /** Instantiate a new AccessibilityPicker.
```

```
        * @param siteName is a valid name of a SabaSite.  This is
        *   by default `SabaWeb' but can be of any name depending
        *   on what is named in SabaAdmin.  This parameter can be
obtained
        *   from the getSiteName() method in a SabaPage.
        * @param orderId is the order id attribute for which we will
        *   be attaching data.
        */
      public AccessibilityPicker(String siteName, String orderId) throws
SabaException {
            super(siteName);
            this.orderId= orderId.trim();
            id = (long)98e13 + System.currentTimeMillis();
      } // AccessibilityPicker

      /*
      private static String delSQL = "delete from fgt_gen where id1=? "
            + " and num1 < ? and id2 like 'listv%'";
      private void remove() throws SQLException, SabaException {
            Connection con = null;
            PreparedStatement ps = null;
            try {
                  con = getConnection();
                  ps = con.prepareStatement(delSQL);
                  log(delSQL);
                  ps.setString(1,orderId);
                  ps.setInt(2,(int)(System.currentTimeMillis()/1000L));
                  log(ps.executeUpdate()+" row(s) deleted.");
            }  finally {
                  if(ps!=null) ps.close();
            } // try finally
      } // remove
      */

      private static String comSQL = "update tpt_oe_order set custom0 =
? "
            + " where id = ?";
      /** setComment() will set (update) the comment and the
        * TPT_OE_ORDER.CUSTOM0 attribute per the named order.
        * @param comment is the comment string that will be placed in
this
        *   field.
        */
      public void setComment(String comment) throws SabaException,
SQLException {
            PreparedStatement ps = null;
            try {
                  ps = getConnection().prepareStatement(comSQL);
                  log(comSQL);
                  ps.setString(1,comment);
                  ps.setString(2,orderId);
                  log(ps.executeUpdate()+" row(s) affected.");
            } finally {
                  if(ps!=null) ps.close();
            } // try finally
      } // setComment

      private static String iacSQL = "insert into fgt_gen "
            + "(id,id1,id2,time_stamp,type,str1) values (?,?,?,?,1,?)";
      /** Set the comment in the FGT_GEN.STR1 attribute per the given
```

```
        * accessibilityId.
        * @param accessibilityId is one of the FGT_LIST_OF_VAL.IDs that
        *   can be obtained through the getAccessibilityIds() method.
        * @param comment is the comment to which to attach to the given
        *   accessibility need.
        */
      public void setAccessibilityComment(String accessibilityId, String
comment) throws SabaException, SQLException {
            PreparedStatement ps = null;
            try {
                  ps = getConnection().prepareStatement(iacSQL);
                  log(iacSQL);
                  ps.setString(1,"acces"+String.valueOf(id++));
                  ps.setString(3,orderId);
                  ps.setString(2,accessibilityId);
                  ps.setString(4,String.valueOf(System.currentTimeMillis
()) );
                  ps.setString(5,comment);
                  log(ps.executeUpdate()+" row(s) inserted.");
            } finally {
                  if(ps!=null) ps.close();
            } // try finally
      } // setAccessibilityComment

      private static String insSQL = "insert into fgt_gen "
            + "(id,id1,id2,time_stamp,type) values (?,?,?,?,1)";
      /** Update the order with the given set (array) of
accessibilityIds.
        * @param accessibilityIds should be a distinct set of
        *   FGT_LIST_OF_VAL.IDs that represent those accessibility
        *   needs that are identified by the user from the
        *   order_receipt.saba page.
        */
      public void insert(Object[] accessibilityIds) throws
SabaException, SQLException {
            // log("Removing...");
            // remove();
            // log("...Removed.");
            PreparedStatement ps = null;
            try {
                  ps = getConnection().prepareStatement(insSQL);
                  log(insSQL);
                  ps.setString(2,orderId);
                  ps.setString(4,String.valueOf(System.currentTimeMillis
()) );
                  // ps.setLong(5,(int)(System.currentTimeMillis
()/1000L));
                  for(int i=0;i<accessibilityIds.length;i++) {
                        ps.setString(1,"acces"+String.valueOf(id++));
                        ps.setString(3,(String)accessibilityIds[i]);
                        log(ps.executeUpdate()+" row(s) inserted.");
                  } // for
            } finally {
                  if(ps!=null) ps.close();
            } // try finally
      } // insert

      private static String locSQL = "select e.id from fgt_list_of_val e
"
            + ",fgt_lov l where e.list_id=l.id and lower(l.name)="
```

```
            + "'accessibility' order by e.id";
      /** Obtain a set of Saba database IDs representing a list of
values
        * for accessibility needs.
        * @return An array of String's that represent the set values
        * and distinct FGT_LIST_OF_VAL.IDs in the database.  These are
        * <b>not<b> the selected accessibilityIds, but rather <b>all</b>
        * possibilities listed in the database for accessibility needs.
        */
      public Object[] getAccessibilityIds() throws SabaException,
SQLException {
            PreparedStatement ps = null;
            ResultSet rs = null;
            Object[] o = null;
            try {
                  ps = getConnection().prepareStatement(locSQL);
                  log(locSQL);
                  rs = ps.executeQuery();
                  ArrayList al = new ArrayList();
                  while(rs.next()) {
                        al.add(rs.getObject(1));
                  } // while
                  o = al.toArray();
            } finally {
                  if(rs!=null) rs.close();
                  if(ps!=null) ps.close();
            } // try finally
            return o;
      } // getAccessibilityIds

      private static String locNSQL = "select name from fgt_list_of_val
"
            + "where id = ?";
      /** Per each obtainable accessibility IDs, obtain the user
interface
        * name for the id.
        * @param id The accessibility id.
        * @return The name associated with the id.
        */
      public String getAccessibilityName(String id) throws
SabaException, SQLException {
            PreparedStatement ps = null;
            ResultSet rs = null;
            String ret = null;
            try {
                  ps = getConnection().prepareStatement(locNSQL);
                  log(locNSQL);
                  ps.setString(1,id.trim());
                  rs = ps.executeQuery();
                  if(rs.next()) ret = rs.getObject(1).toString();
            } finally {
                  if(rs!=null) rs.close();
                  if(ps!=null) ps.close();
            } // try finally
            return ret;
      } // getLocationName

      /** Unit test method. */
      public static void main(String[] argv) throws Throwable {
            // Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
            // Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@10.10.60.167:1521:DEMO4x","tp2","tp2");
            String id = "intor000000000001040";
            id = argv[0];
            AccessibilityPicker ap = new AccessibilityPicker
("SabaWeb",id);
            Object[] o = ap.getAccessibilityIds();
            ArrayList al = new ArrayList();
            for(int i=0;i<o.length;i++) {
                  System.out.println(o[i]+":"+ap.getAccessibilityName
((String)o[i]) );
                  al.add(o[i]);
                  ap.setAccessibilityComment(o[i].toString(),o
[i].toString());
            }
            ap.insert(al.toArray());
            ap.setComment("This is quite interesting.");

            ap.close();
            // Logger.getWriter().flush();
      }  // main

} // LocationsPicker
```

```
delete from fgt_lov where id like 'listi98%';
insert into fgt_lov(id,flags,name,time_stamp,description) values (
      'listi9800000000000001',
      '0000000000',
      'accessibility',
      '1011126762000',
      'Accessibility Needs'
);

delete from fgt_list_of_val where id like 'listv98%';
insert into fgt_list_of_val
(id,list_id,locale_id,category,enabled,name,time_stamp) values (
      'listv9800000000000001',
      'listi9800000000000001',
      'local000000000000001',
      0,
      1,
      'Deafness',
      '1011126762000'
);

insert into fgt_list_of_val
(id,list_id,locale_id,category,enabled,name,time_stamp) values (
      'listv9800000000000002',
      'listi9800000000000001',
      'local000000000000001',
      0,
      1,
      'Blindness',
      '1011126762000'
);

insert into fgt_list_of_val
(id,list_id,locale_id,category,enabled,name,time_stamp) values (
      'listv9800000000000003',
      'listi9800000000000001',
      'local000000000000001',
      0,
      1,
      'Physical Disability',
      '1011126762000'
);


insert into fgt_list_of_val
(id,list_id,locale_id,category,enabled,name,time_stamp) values (
      'listv9800000000000004',
      'listi9800000000000001',
      'local000000000000001',
      0,
      1,
      'Other',
      '1011126762000'
);
```

All Classes

AccessibilityPicker
SabaObject
User

How This API Document Is Organized

This API (Application Programming Interface) document has pages corresponding to the items in the navigation bar, described as follows.

Overview

The Overview page is the front page of this API document and provides a list of all packages with a summary for each. This page can also contain an overall description of the set of packages.

Package

Each package has a page that contains a list of its classes and interfaces, with a summary for each. This page can contain four categories:

- Interfaces (italic)
- Classes
- Exceptions
- Errors

Class/Interface

Each class, interface, inner class and inner interface has its own separate page. Each of these pages has three sections consisting of a class/interface description, summary tables, and detailed member descriptions:

- Class inheritance diagram
- Direct Subclasses
- All Known Subinterfaces
- All Known Implementing Classes
- Class/interface declaration
- Class/interface description

- Inner Class Summary
- Field Summary
- Constructor Summary
- Method Summary

- Field Detail
- Constructor Detail
- Method Detail

Each summary entry contains the first sentence from the detailed description for that item. The summary entries are alphabetical, while the detailed descriptions are in the order they

appear in the source code. This preserves the logical groupings established by the programmer.

Tree (Class Hierarchy)

There is a Class Hierarchy page for all packages, plus a hierarchy for each package. Each hierarchy page contains a list of classes and a list of interfaces. The classes are organized by inheritance structure starting with `java.lang.Object`. The interfaces do not inherit from `java.lang.Object`.

- When viewing the Overview page, clicking on "Tree" displays the hierarchy for all packages.
- When viewing a particular package, class or interface page, clicking "Tree" displays the hierarchy for only that package.

Deprecated API

The Deprecated API page lists all of the API that have been deprecated. A deprecated API is not recommended for use, generally due to improvements, and a replacement API is usually given. Deprecated APIs may be removed in future implementations.

Index

The Index contains an alphabetic list of all classes, interfaces, constructors, methods, and fields.

Prev/Next

These links take you to the next or previous class, interface, package, or related page.

Frames/No Frames

These links show and hide the HTML frames. All pages are available with or without frames.

Serialized Form

Each serializable or externalizable class has a description of its serialization fields and methods. This information is of interest to re-implementors, not to developers using the API. While there is no link in the navigation bar, you can get to this information by going to any serialized class and clicking "Serialized Form" in the "See also" section of the class description.

This help file applies to API documentation generated using the standard doclet.

com.saba.denver

Class User

```
java.lang.Object
  |
  +--com.saba.denver.SabaObject
        |
        +--com.saba.denver.User
```

public class **User**
extends SabaObject

User represents a generic user in the system, be that a person or an employee.

Constructor Summary

User(java.lang.String siteName, java.lang.String username) Instantiate a new User object referencing a given user in the Saba database.

Method Summary

java.lang.String	**getPassword**() Obtain the *cleartext* password for this User.
static void	**main**(java.lang.String[] argv) Unit test method.

Methods inherited from class com.saba.denver.SabaObject
close, finalize, getConnection, log

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

User

```
public User(java.lang.String siteName,
            java.lang.String username)
```

Instantiate a new User object referencing a given user in the Saba database.

Parameters:

`siteName` - The Saba site's name from each page. This can be obtained via the getSiteName () method in the SabaPage.

`username` - The username (not id) in the database for the given user.

Method Detail

getPassword

```
public java.lang.String getPassword()
                              throws java.sql.SQLException
```

Obtain the *cleartext* password for this User.

---

main

```
public static void main(java.lang.String[] argv)
                 throws java.lang.Throwable
```

Unit test method.

Deprecated API

Overview Package Class **Tree** **Deprecated** **Index** **Help**
PREV NEXT **FRAMES** **NO FRAMES**

```
import java.sql.*;
import java.io.*;

public class Describe {

      public static void main(String[] argv) throws Throwable {

            Class.forName("oracle.jdbc.driver.OracleDriver");
            String url = "jdbc:oracle:thin:@emsdb1:1521:emsdb6";
            String user = "tp2";
            String pass = user;

            PrintWriter pw = new PrintWriter(System.out,true);

            Connection con = DriverManager.getConnection(url,user,pass);
            String sql = "select * from "+argv[0];
            Statement s = con.createStatement();
            ResultSet rs = s.executeQuery(sql);
            pw.println(argv[0].toUpperCase());
            pw.println("--");
            if(rs.next()) {
                  ResultSetMetaData md = rs.getMetaData();
                  for(int i=1;i<=md.getColumnCount();i++) {
                        // pw.print(md.getColumnLabel(i)+"\t");
                        pw.print(md.getColumnName(i)+"\t");
                        pw.print(md.getColumnTypeName(i)+"\t");
                        pw.print((md.isNullable(i)
==md.columnNullable?"NULL":"NOT NULL")+"\t");
                        pw.println(md.getPrecision(i)+"."+md.getScale
(i));
                  } // for
            } // if

            pw.close();
            rs.close();
            s.close();
            con.close();
      } // main

} // Describe
```

A

C

F

G

I

insert(Object[]) - Method in class com.saba.doe.AccessibilityPicker
Update the order with the given set (array) of accessibilityIds.

L

log(Object) - Method in class com.saba.denver.SabaObject
Any subclass may call log() to log out to the standard error file, most often found in jrun/jsm-default/logs/stderr.log.

M

main(String[]) - Static method in class com.saba.doe.AccessibilityPicker
Unit test method.
main(String[]) - Static method in class com.saba.denver.User
Unit test method.

S

SabaObject - class com.saba.denver.SabaObject.
SabaObject is a basic low-level class that abstracts the getting and setting of Saba database connections (given a particular Saba site).
SabaObject(String) - Constructor for class com.saba.denver.SabaObject
Instantiate a new SabaObject and make sure it relates itself to a particular Saba site.
setAccessibilityComment(String, String) - Method in class com.saba.doe.AccessibilityPicker
Set the comment in the FGT_GEN.STR1 attribute per the given accessibilityId.
setComment(String) - Method in class com.saba.doe.AccessibilityPicker
setComment() will set (update) the comment and the TPT_OE_ORDER.CUSTOM0 attribute per the named order.

U

User - class com.saba.denver.User.
User represents a generic user in the system, be that a person or an employee.
User(String, String) - Constructor for class com.saba.denver.User
Instantiate a new User object referencing a given user in the Saba database.

This set of source files represent the added logic that is developed for Accenture to represent and better manage customization needs for the SFA/DOE project. All table structures for the data for accessibility are documented in the com.saba.doe.AccessibilityPicker class documentation.

See:

Description

Packages	
com.saba.denver	
com.saba.doe	

This set of source files represent the added logic that is developed for Accenture to represent and better manage customization needs for the SFA/DOE project.

All table structures for the data for accessibility are documented in the com.saba.doe.AccessibilityPicker class documentation. Accessibility is related through an Entitry Relationship (ER) table in Saba called the FGT_GEN table. Normally, relations will exist in the tables (obejcts) themselves, however we are rlating list of value (LOV) items to an order, which is not common, thus we will store the database ID of the order and a list of values item database ID per row for each accessibility need (list of values ID) that is represented by the user that they themselves need per order.

Accessibility Needs will be attached to orders and NOT to user profiles giving a much more flexible representation of accessibility needs per class instead of globally per learner.

The tables used in storing Accessibility information consist of four tables:

- FGT_GEN,
- FGT_LOV,
- FGT_LIST_OF_VAL, and
- TPT_OE_ORDER.

At the very root of accessibility needs, the needs themselves are stored as a ``list of values.'' This list of values contains an arbitrary `list' of items such as Deafness, Blindness, et cetera.

The list of values themselves (the arbitrary list itself) must have a name. Since all list of values items (Blindness, Deafness, etc) are *all* stored in the same table with an arbitrary number of other list of values, a unique name for this list istelf is effectively a reference creating a subset of the list appropriate for accessibility needs.

The encompassing name for any given list is stored in the table FGT_LOV. The name of the list is ``Accessibility Needs,'' and stored in the attribute FGT_LOV.NAME. An ID (FGT_LOV.ID) can be obtained from this table where the name (FGT_LOV.NAME) is ``Accessibility Needs.'' The ID obtained from this table will be prefixed by `listi.'

Each of the items for the list of values named by ``Accessibility Needs'' (Blindness, Deafness, etc) is stored in the table FGT_LIST_OF_VAL and each has a unique row. Each row in this table has a unique ID that is prefixed by `listv.' The attribute FGT_LIST_OF_VAL.LIST_ID is a foreign key marker to the actual list name ID and links to FGT_LOV.ID.

Every order placed in the system (regardless of how many order items there are associated with every order) is stored in a unique row in TPT_OE_ORDER. Each row in this table will either have the prefix `intor' or `extor' for internal orders and external orders, respectfully.

As we wish to associate accessibility needs with each order (internal or external), Saba contains a very *generic* Entity Relationship (ER) table called FGT_GEN. In order to associate orders with not only a list of values but each selected list of value items, individual rows will be inserted into FGT_GEN to create the relationship that otherwise does not naturally exist in the Saba database schema (mapping orders to accessibility list of value items).

For every relationship that is made between orders and accessibility needs, a multitude of rows will be inserted into FGT_GEN. All of these rows will be prefixed with the ID prefix of `acces.'

To relate a single list of values item (Blindness or Deafness, etc) to a particular order, a new row will be inserted into FGT_GEN with three primary fields being populated. FGT_GEN.ID is the unique ID for the row (acces); FGT_GEN.ID1 will contain the order ID of the order in question (intor, extor); FGT_GEN.ID2 will contain the list of values item (listv).

For each selected list of values item, a single row matching the above description is inserted. Each list of values item can have a separate text field associated with it. In this case, the FGT_GEN table will have FGT_GEN.ID1 with the list of values item (listv) ID, the FGT_GEN.ID2 will have the order ID (intor, extor), and the FGT_GEN.STR1 will contain the text associated with each test entry for selected list of values items.

A general comment may be made on the accessibility needs for each order. This general overall comment is updated in the TPT_OE_ORDER.CUSTOM0 field.

This set of source files represent the added logic that is developed for Accenture to represent and better manage customization needs for the SFA/DOE project.

All table structures for the data for accessibility are documented in the com.saba.doe.AccessibilityPicker class documentation. Accessibility is related through an Entitry Relationship (ER) table in Saba called the FGT_GEN table. Normally, relations will exist in the tables (obejcts) themselves, however we are rlating list of value (LOV) items to an order, which is not common, thus we will store the database ID of the order and a list of values item database ID per row for each accessibility need (list of values ID) that is represented by the user that they themselves need per order.

Accessibility Needs will be attached to orders and NOT to user profiles giving a much more flexible representation of accessibility needs per class instead of globally per learner.

The tables used in storing Accessibility information consist of four tables:

- FGT_GEN,
- FGT_LOV,
- FGT_LIST_OF_VAL, and
- TPT_OE_ORDER.

At the very root of accessibility needs, the needs themselves are stored as a ``list of values.'' This list of values contains an arbitrary `list' of items such as Deafness, Blindness, et cetera.

The list of values themselves (the arbitrary list itself) must have a name. Since all list of values items (Blindness, Deafness, etc) are *all* stored in the same table with an arbitrary number of other list of values, a unique name for this list istelf is effectively a reference creating a subset of the list appropriate for accessibility needs.

The encompassing name for any given list is stored in the table FGT_LOV. The name of the list is ``Accessibility Needs,'' and stored in the attribute FGT_LOV.NAME. An ID (FGT_LOV.ID) can be obtained from this table where the name (FGT_LOV.NAME) is ``Accessibility Needs.'' The ID obtained from this table will be prefixed by `listi.'

Each of the items for the list of values named by ``Accessibility Needs'' (Blindness, Deafness, etc) is stored in the table FGT_LIST_OF_VAL and each has a unique row. Each row in this table has a unique ID that is prefixed by `listv.' The attribute FGT_LIST_OF_VAL.LIST_ID is a foreign key marker to the actual list name ID and links to FGT_LOV.ID.

Every order placed in the system (regardless of how many order items there are associated with every order) is stored in a unique row in TPT_OE_ORDER. Each row in this table will either have the prefix `intor' or `extor' for internal orders and external orders, respectfully.

As we wish to associate accessibility needs with each order (internal or external), Saba contains a very *generic* Entity Relationship (ER) table called FGT_GEN. In order to associate orders with not only a list of values but each selected list of value items, individual rows will be inserted into FGT_GEN to create the relationship that otherwise does not naturally exist in the Saba database schema (mapping orders to accessibility list of value items).

For every relationship that is made between orders and accessibility needs, a multitude of rows will be inserted into FGT_GEN. All of these rows will be prefixed with the ID prefix of `acces.'

To relate a single list of values item (Blindness or Deafness, etc) to a particular order, a new row will be inserted into FGT_GEN with three primary fields being populated. FGT_GEN.ID is the unique ID for the row (acces); FGT_GEN.ID1 will contain the order ID of the order in question (intor, extor); FGT_GEN.ID2 will contain the list of values item (listv).

For each selected list of values item, a single row matching the above description is inserted. Each list of values item can have a separate text field associated with it. In this case, the FGT_GEN table will have FGT_GEN.ID1 with the list of values item (listv) ID, the FGT_GEN.ID2 will have the order ID (intor, extor), and the FGT_GEN.STR1 will contain the text associated with each test entry for selected list of values items.

A general comment may be made on the accessibility needs for each order. This general overall comment is updated in the TPT_OE_ORDER.CUSTOM0 field.

This set of source files represent the added logic that is developed for Accenture to represent and better manage customization needs for the SFA/DOE project. All table structures for the data for accessibility are documented in the com.saba.doe.AccessibilityPicker class documentation.

See:

Description

Packages	
com.saba.denver	
com.saba.doe	

This set of source files represent the added logic that is developed for Accenture to represent and better manage customization needs for the SFA/DOE project.

All table structures for the data for accessibility are documented in the com.saba.doe.AccessibilityPicker class documentation. Accessibility is related through an Entitry Relationship (ER) table in Saba called the FGT_GEN table. Normally, relations will exist in the tables (obejcts) themselves, however we are rlating list of value (LOV) items to an order, which is not common, thus we will store the database ID of the order and a list of values item database ID per row for each accessibility need (list of values ID) that is represented by the user that they themselves need per order.

Accessibility Needs will be attached to orders and NOT to user profiles giving a much more flexible representation of accessibility needs per class instead of globally per learner.

The tables used in storing Accessibility information consist of four tables:

- FGT_GEN,
- FGT_LOV,
- FGT_LIST_OF_VAL, and
- TPT_OE_ORDER.

At the very root of accessibility needs, the needs themselves are stored as a ``list of values.'' This list of values contains an arbitrary `list' of items such as Deafness, Blindness, et cetera.

The list of values themselves (the arbitrary list itself) must have a name. Since all list of values items (Blindness, Deafness, etc) are *all* stored in the same table with an arbitrary number of other list of values, a unique name for this list istelf is effectively a reference creating a subset of the list appropriate for accessibility needs.

The encompassing name for any given list is stored in the table FGT_LOV. The name of the list is ``Accessibility Needs,'' and stored in the attribute FGT_LOV.NAME. An ID (FGT_LOV.ID) can be obtained from this table where the name (FGT_LOV.NAME) is ``Accessibility Needs.'' The ID obtained from this table will be prefixed by `listi.'

Each of the items for the list of values named by ``Accessibility Needs'' (Blindness, Deafness, etc) is stored in the table FGT_LIST_OF_VAL and each has a unique row. Each row in this table has a unique ID that is prefixed by `listv.' The attribute FGT_LIST_OF_VAL.LIST_ID is a foreign key marker to the actual list name ID and links to FGT_LOV.ID.

Every order placed in the system (regardless of how many order items there are associated with every order) is stored in a unique row in TPT_OE_ORDER. Each row in this table will either have the prefix `intor' or `extor' for internal orders and external orders, respectfully.

As we wish to associate accessibility needs with each order (internal or external), Saba contains a very *generic* Entity Relationship (ER) table called FGT_GEN. In order to associate orders with not only a list of values but each selected list of value items, individual rows will be inserted into FGT_GEN to create the relationship that otherwise does not naturally exist in the Saba database schema (mapping orders to accessibility list of value items).

For every relationship that is made between orders and accessibility needs, a multitude of rows will be inserted into FGT_GEN. All of these rows will be prefixed with the ID prefix of `acces.'

To relate a single list of values item (Blindness or Deafness, etc) to a particular order, a new row will be inserted into FGT_GEN with three primary fields being populated. FGT_GEN.ID is the unique ID for the row (acces); FGT_GEN.ID1 will contain the order ID of the order in question (intor, extor); FGT_GEN.ID2 will contain the list of values item (listv).

For each selected list of values item, a single row matching the above description is inserted. Each list of values item can have a separate text field associated with it. In this case, the FGT_GEN table will have FGT_GEN.ID1 with the list of values item (listv) ID, the FGT_GEN.ID2 will have the order ID (intor, extor), and the FGT_GEN.STR1 will contain the text associated with each test entry for selected list of values items.

A general comment may be made on the accessibility needs for each order. This general overall comment is updated in the TPT_OE_ORDER.CUSTOM0 field.

Hierarchy For All Packages

Package Hierarchies:
com.saba.denver, com.saba.doe

Class Hierarchy

- class java.lang.Object
  - class com.saba.denver.**SabaObject**
    - class com.saba.doe.**AccessibilityPicker**
    - class com.saba.denver.**User**

Package com.saba.doe

Class Summary	
AccessibilityPicker	AccessibilityPicker is a custom class that will manage the data for the DOE/SFA/Accenture Accessibility needes.

Overview **Package** Class **Tree** **Deprecated** **Index** **Help**
PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES

Hierarchy For Package com.saba.doe

Package Hierarchies:
All Packages

Class Hierarchy

- class java.lang.Object
  - class com.saba.denver.**SabaObject**
    - class com.saba.doe.**AccessibilityPicker**

com.saba.denver

Class SabaObject

```
java.lang.Object
  |
  +--com.saba.denver.SabaObject
```

Direct Known Subclasses:
AccessibilityPicker, User

---

public abstract class **SabaObject**
extends java.lang.Object

SabaObject is a basic low-level class that abstracts the getting and setting of Saba database connections (given a particular Saba site). To obtain a Saba database connection in a class, extend this class and you will be able to use the getConnection() method to obtain said connection. This Connection object is *not* able to be closed. One will instead return it to the SabaObject by calling the freeConnection() method, which will return this object to the connection pool.

See Also:
`Connection`

Constructor Summary

SabaObject(java.lang.String siteName) Instantiate a new SabaObject and make sure it relates itself to a particular Saba site.

Method Summary

void	**close**() In any usage of SabaObject, try{}finally{} any use and be absolutely sure that in your finally{} block you close() this Object.
void	**finalize**() Just in case someone does *not* call close, please close the Connection object out upon GC.
java.sql.Connection	**getConnection**() Obtain a Saba Connection.
protected void	**log**(java.lang.Object o) Any subclass may call log() to log out to the standard error file, most often

	found in jrun/jsm-default/logs/stderr.log.

Methods inherited from class java.lang.Object
`clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

SabaObject

```
public SabaObject(java.lang.String siteName)
```

Instantiate a new SabaObject and make sure it relates itself to a particular Saba site.

Parameters:

`siteName` - The Saba site to be processed. Obtained through the getSiteName() method call in a SabaPage.

See Also:

`com.saba.page.SabaPage`

Method Detail

log

```
protected void log(java.lang.Object o)
```

Any subclass may call log() to log out to the standard error file, most often found in jrun/jsm-default/logs/stderr.log. This will write the current date, and thread information and the toString() representation of the passed Object.

Parameters:

`o` - The object to represent a s a string.

See Also:

`Date`, `java.lang.Thread`, `Object`

---

getConnection

```
public final java.sql.Connection getConnection()
                                        throws java.sql.SQLException,
                                               com.saba.exception.SabaException
```

Obtain a Saba Connection. This Connection object should *never* be close()d, but rather be returned to SabaObject via the freeConnection() method.

Returns:

A Saba database Connection.

See Also:

`Connection`

---

close

```
public final void close()
                throws com.saba.exception.SabaException
```

In any usage of SabaObject, try{}finally{} any use and be absolutely sure that in your finally{} block you close() this Object.

---

finalize

```
public void finalize()
```

Just in case someone does *not* call close, please close the Connection object out upon GC.

Overrides:

finalize in class java.lang.Object

```
/* $Id:$ */

package com.saba.denver;

import java.sql.*;

import com.saba.db.*;
import com.saba.exception.*;

/** SabaObject is a basic low-level class that abstracts the getting
  * and setting of Saba database connections (given a particular Saba
  * site).  To obtain a Saba database connection in a class, extend this
  * class and you will be able to use the getConnection() method to
obtain
  * said connection.  This Connection object is <i>not</i> able to be
closed.
  * One will instead return it to the SabaObject by calling the
freeConnection()
  * method, which will return this object to the connection pool.
  * @see java.sql.Connection
  * @author jsjensen@saba.com
  */
public abstract class SabaObject {

      private String siteName;
      private Connection con;
      private Object connectionLock;

      /** Instantiate a new SabaObject and make sure it relates itself
        * to a particular Saba site.
        * @param siteName The Saba site to be processed.  Obtained
through
        *   the getSiteName() method call in a SabaPage.
        * @see com.saba.page.SabaPage
        */
      public SabaObject(String siteName) {
            this.siteName = siteName;
            connectionLock = new Object();
      }

      /** Any subclass may call log() to log out to the standard error
file,
        * most often found in jrun/jsm-default/logs/stderr.log.  This
will
        * write the current date, and thread information and the
toString()
        * representation of the passed Object.
        * @see java.util.Date
        * @see java.lang.Thread
        * @see java.lang.Object
        * @param o The object to represent a s a string.
        */
      protected void log(Object o) {
            System.err.println(
                  new java.util.Date().toString() + " [" +
                  Thread.currentThread().toString() + "] " +
                  o.toString()
            );
      }
```

```
      /** Obtain a Saba Connection.   This Connection object should <i>
        * never </i> be close()d, but rather be returned to SabaObject
via
        * the freeConnection() method.
        * @return A Saba database Connection.
        * @see java.sql.Connection
        */
      final public Connection getConnection() throws SQLException,
SabaException {
            synchronized(connectionLock) {
                  if(con==null) {
                        con = DbConnectionPoolManager.getConnection
(siteName);
                        con.setAutoCommit(true);
                  } // if
                  log("Obtained "+con.toString());
                  return con;
            } // sync
      } // getConnection

      /** In any usage of SabaObject, try{}finally{} any use and be
absolutely
        * sure that in your finally{} block you close() this Object.
        */
      final public void close() throws SabaException {
            synchronized(connectionLock) {
                  if(con!=null) {
                        String s = con.toString();
                        DbConnectionPoolManager.freeConnection
(siteName,con);
                        con = null;
                        log("Freeing "+s);
                  }
            } // sync
      }

      /** Just in case someone does <i>not</i> call close, please close
        * the Connection object out upon GC.
        */
      public void finalize() {
            try {
                  close();
            } catch (Exception ignore) {}
      } // finalize

} // SabaObject
```

Serialized Form

Overview Package Class **Tree** **Deprecated** **Index** **Help**
PREV NEXT **FRAMES** **NO FRAMES**

```
/* $Id:$ */

/* Copyright (C) 2002   Saba Software, Services, J. S. Jensen
 *    mailto:jsjensen@saba.com
 * All rights are non-exclusive.
 */
package com.saba.denver;

import java.sql.*;

/** User represents a generic user in the system, be that a person
  * or an employee.
  */
public class User extends SabaObject {

      private String username;

      /** Instantiate a new User object referencing a given user in the
        * Saba database.
        * @param siteName The Saba site's name from each page.  This can
be
        *   obtained via the getSiteName() method in the SabaPage.
        * @param username The username (not id) in the database for the
given
        *   user.
        */
      public User(String siteName, String username) {
            super(siteName);
            // Saba stores usernames in uppercase.
            this.username = username.trim().toUpperCase();
      }

      private static String sql = "select password from tpt_employees
where username = ? "
            + " union select password from tpt_person where username = ?
";
      /** Obtain the <i>cleartext</i> password for this User. */
      public String getPassword() throws SQLException {
            PreparedStatement ps = null;
            ResultSet rs = null;
            String passwd = "";
            try {
                  ps = getConnection().prepareStatement(sql);
                  ps.setString(1,username);
                  ps.setString(2,username);
                  rs = ps.executeQuery();
                  if(rs.next()) passwd = rs.getString(1);
            } catch (Exception e) {
                  log(e);
            } finally {
                  if(rs!=null) rs.close();
                  if(ps!=null) ps.close();
            } // try catch finally
            System.err.println("usernm="+username);
            System.err.println("passwd="+passwd);
            String cleartext = com.saba.security.SabaLogin.decrypt
(passwd);
            return cleartext;
      } // getPassword
```

```
	/** Unit test method. */
	public static void main(String[] argv) throws Throwable {
		User u = null;
		try {
			u = new User("SabaWeb","ketand");
			System.out.println(u.getPassword());
		} finally {
			// if(u!=null) u.close();
		} // try finally

		u = null;
		Thread.sleep(5000);
		System.gc();
	} // main

} // User
```